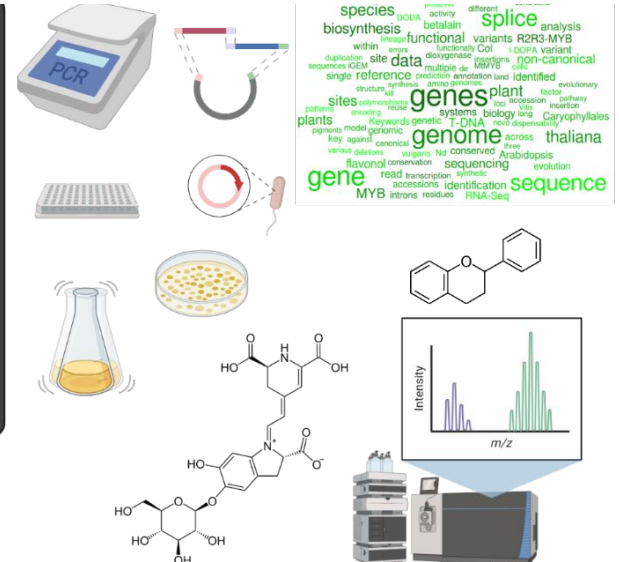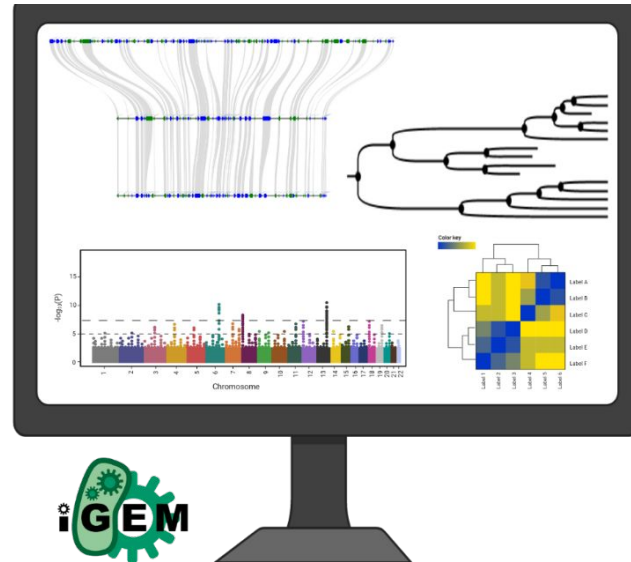# Python - Getting started

Prof. Dr. Boas Pucker (Plant Biotechnology and Bioinformatics)

Availability of slides

- All materials are freely available (CC BY) - after the lectures:
  - StudIP: 'Python for Life Scientists'
  - GitHub: https://github.com/bpucker/teaching

- Questions: Feel free to ask at any time

- Feedback, comments, or questions: b.pucker[a]tu-bs.de

My figures and content can be re-used in accordance with CC BY 4.0, but this might not apply to all images/logos. Some figure were constructed using bioRender.com.

Technische Universität Braunschweig

Installing Jupyter Notebook

- Linux:
    - $ sudo apt update
    - $ sudo apt install python3-pip python3-dev
    - $ mkdir python_course
    - $ cd python_course
    - $ virtualenv python_course
    - $ source python_course/bin/activate
    - $ pip install jupyter
    - $ jupyter notebook
- Windows:
    - 1) Install Anaconda
      (https://www.geeksforgeeks.org/how-to-install-anaconda-on-windows/)
    - 2) Install Python3 through Anaconda
    - 3) Install Jupyter through Anaconda
      (https://www.geeksforgeeks.org/how-to-install-jupyter-notebook-in-windows/)
- Mac:
    - See instructions above

Technische
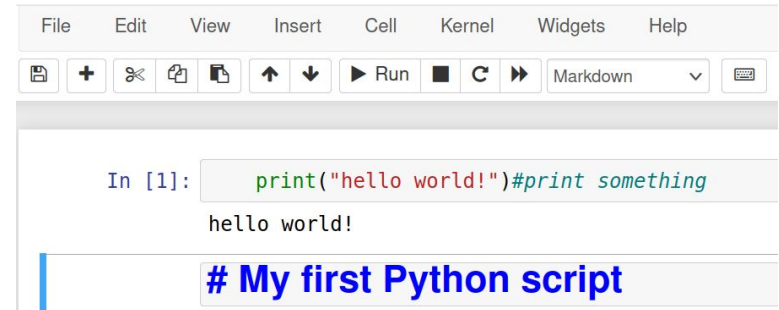Universität
Braunschweig

Starting Jupyter Notebook

- Start Jupyter Notebook

- Create a new file:
  - 'New'
  - Python3 ipykernel
  - Change name

- Python code is written and executed in Jupyter Notebook

Technische
Universität
Braunschweig
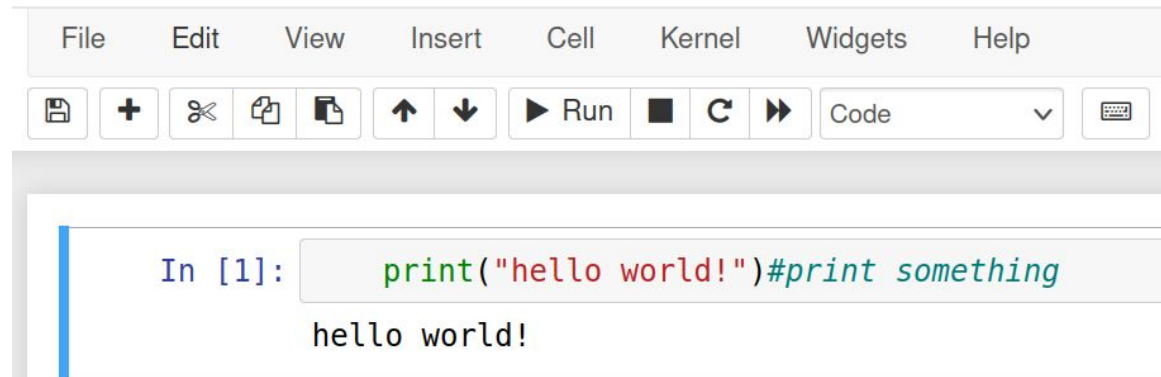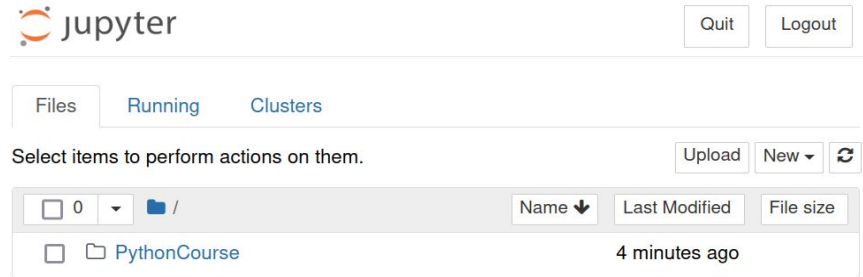
Adding elements to a Jupyter Notebook

- Code: you know this already

- Header: structure your work/documentation

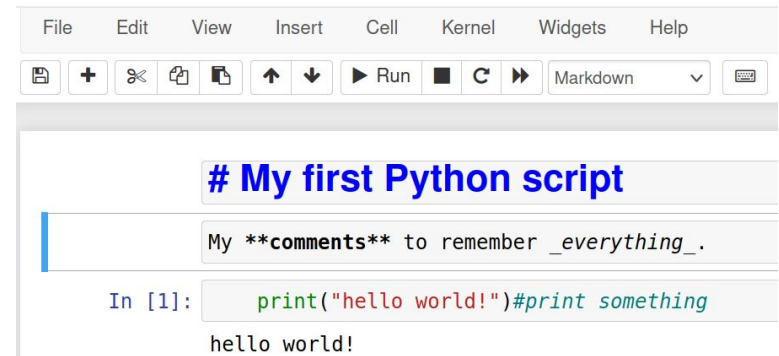- Markdown: comments with specific formatting

Simple commands & variable types

- print() allows you to show the results of calculations in the terminal

- Different types of variables can be stored in a list

- Different types of number (int or float)

```python
print("test") #print
my_list = ["1", "2", "3", 4, 5, 6 ] #list
my_int = 1 #int
my_float = float(3.1) #float
my_string = str(my_int) #string
```

Comments and structure

- Two ways to add comments:

- '#' rest of the line is comment and ignored by Python

- Triple quotation marks allow comments over multiple lines

- Use ASCII characters only (**no** ä, ö, ü, ß, …)

- Empty lines are ignored by Python (use space to structure code)

Assignment and comparison

- '='used to assign value to variable

- '==' compares two values/variables

- Variable names may contain characters, underline, and numbers (not at the start)

```
In [4]: a = 'hello world!'
        print(a)

        hello world!

In [5]: b = 'test'
        c = 'test'
        a == b

Out[5]: False

In [6]: b == c

Out[6]: True
```

Variable type: string

- a, b, c are strings ('str')

- Python allows to check the variable type:
  - type(a)

- Almost all variable types can be converted to string:
  - str(<VARIABLE>)

Technische
Universität
Braunschweig

Variable types: integer & float

- Two variable types for numbers
  - Integer = complete number (example: 3)
  - Float = decimal number (example: 3.1415926)

- Important: '.' NOT ',' separates numbers in float

- Some strings can be converted to integer/float
  - My_int = int('3')
  - My_float = float('3.145926')

- Check result via type(<VARIABLE>)

Technische
Universität
Braunschweig

Python as calculator

- Numbers can be used for calculations
  - a = 3
  - b = 2
  - print(a+b)
  - print(a*b)
  - print(a/b)
  - print(a/float(b))
  - print(a%b)
  - print(a<b)
  - print(a!=b)

- How to calculate roots?

- Interested in more complex math? (NumPy, SciPy)

Variable type: list

- List can contain elements of different types (e.g. strings)

- Elements can be accessed via index

- Index is given in square brackets after the list name:

- Matching your expectation?

```python
my_list = [ "one", "two", "three" ]
print( my_list[1] )
```

Indices in Python

- Python starts counting at 0!!!

```python
my_list = ["one","two","three"]
#          0      1      2
```

- Lists can be concatenated

```python
new_list = new_list + ["four", "five", "six", "seven"]
#new_list = ["one","two","three", "four", "five", "six", "seven"]
#            0      1      2        3       4       5      6
```

- Print subset of list

```python
print(new_list[3:])
print(new_list[:3])
print(new_list[3:5])
```

- Two indices: index1=first element to include; index2=first element following the selection

Technische
Universität
Braunschweig

Indices in Python II

- Strings have indices as well

- -1 points to the last element of a string/list

- -5 points to the 5th element from the end of a string/list

```python
a = 'hello world test string!'
print(a[1:])
print(a[5:10])
print(a[:-1])
print(a[-5:-1])
```

Variable type: boolean (True/False)

- Already used for comparison:

```python
print(1==1)
print(1>1)
print(1==True)
print(True+True)
print(True+False)
```

- Boolean variables can be used for calculations (like numbers)

- Most of the time used only for internal calculations

Brackets

- Two important types of brackets
  - '[]' to generate lists and to access elements via index
  - '()' to transfer arguments to functions

```
a = []
b = "test"
print(b[1])
```

- What are functions?
  - Examples:

```
x = 3.145
str(x)
int(x)
float(x)
```

Technische
Universität
Braunschweig

Exercises - Part1a

- 1.1) Save 3,14159265359 in a variable of type float!

- 1.2) Convert variable from float to integer!

- 1.3) Convert variable back! What happens?

- 1.4) Convert variable to type string!

- 1.5) Save 'Python' in a string variable!

- 1.6) Convert variable type to float! What happens?

- 1.7) What is a pitfall in regard to division when working with int/float?

Connecting strings / lists

- Two lists can be concatenated by using the '+' operator

```python
my_list1 = ["one", "two"]
my_list2 = ["3", "4"]
merged_list = my_list1 + my_list2
print(merged_list)
```

```
['one', 'two', '3', '4']
```

- Two strings can be concatenated by using the '+' operator

```python
my_string1 = "hello"
my_string2 = "user"
merged_string = my_string1 + " " + my_string2
print(merged_string)
```

```
hello user
```

- Mixing different variable types does not work!

Technische
Universität
Braunschweig

Adding content to lists - append()

- Individual elements can be added to lists in a more efficient way:

```python
THE_list = ["1", "2", "3"]
THE_list.append("4")
print(THE_list)
```

```
['1', '2', '3', '4']
```

Exercises - Part1b

- 1.8) Build a sentence based on individual strings!

- 1.9) Find the most efficient way to build this string:
  "hi, user!hi, user!hi, user!hi, user!hi, user!hi, user!hi, user!hi, user!hi, user!hi, user!"

Time for questions!